BIGDATA Processing with Hadoop
(PIG Scripting)

Problem 1: Computing Bigram Statistics

Each line in a file has the following format:

bigram TAB **year** TAB **match_count** TAB **volume_count** NEWLINE

An example for 2-grams (or bigram) would be:

I am 1936 342 90
I am 1945 211 10
very cool 1923 500 10
very cool 1980 3210 1000
very cool 2012 9994 3020

This tells us that, in 1936, the bigram 'I am' appeared 342 times in 90 distinct volumes (books). In 1945, 'I am' appeared 211 times in 10 distinct volumes. And so on.

Write PIG Scripts to compute the following:

1. For each unique bigram, compute its average number of appearances per year of occurrence (we won't be considering volume_count at all). For the above example, the results will be the following:

I am (342 + 211) / 2 = **276.5**
very cool (500 + 3210 + 9994) / 3 = **1568**

2. Output the 5 bigrams with the highest average number of appearances per year of occurrence along with their corresponding average sorted in the descending order. If multiple bigrams with the same average, write down any ones that you like (that is, break ties as you wish). For the above example, the output will be the following (the output can be comma-separated as shown below or tab-separated):

very cool,**1568**
I am,**276.5**

Problem 2: Finding Common friends among non-friends

Please create a pig script `friends.pig` that implements LinkedIn's use of Hadoop to determine how many friends two non-friends share in common. The input is a list of comma separated contact pairs in a file `friends.txt`, e.g.,

```
Alva,Mark
Siri,Mark
Alva,George
Siri,George
Fred,George
```

BIGDATA Processing with Hadoop (PIG Scripting)

The output of your script should be a set of sentences describing how many friends two non-friends have in common, e.g.,

```
(Alva and Siri have 2 friend(s) in common)
(Fred and Siri have 1 friend(s) in common)
```

where every pair reported in the output are not already friends in the input. You should list these so that the pairs are in alphabetical order, and so that listing is in order according to number of friends, first name, and second name.

Assumption: The friend relationship is symmetric, i.e., listing `Alva Mark` implicitly means that `Mark Alva` as well.

Problem 3: Basic Statistics for Twitter Dataset

We will work on a Twitter dataset that was obtained from this link: http://an.kaist.ac.kr/traces/WWW2010.html

The format of the dataset (both local and cluster) is the following:

```
USER_ID \t FOLLOWER_ID \n
```

- USER_ID and FOLLOWER_ID are represented by numeric ID (integer).
- These numeric IDs are the same as numeric IDs Twitter managed.

```
Example:
12      13
12      14
12      15
16      17
```

- Users 13, 14 and 15 are followers of user 12.
- User 17 is a follower of user 16.

Write PIG scripts to compute the following quantities:

a) For each user, calculate the total number of followers of that user. The output format should be like: `USER_ID \t No. of FOLLOWERs \n`

b) Find all the two-hop social relations in the Twitter network. For example, if user 12 is followed by user 13 and user 13 is followed by user 19, then there is a two-hop relation between user 12 and user 19. The output format should be like:
`USER_1_ID \t USER_2_ID`

Problem 4: Finding users who didn't logout

Write PIG script `lazy.pig` that determines which users are forgetting to log out of their accounts. The input schema is:
`log: {time:long, user:chararray, action:chararray, status:chararray, address:chararray}` where

- time is the seconds since the Epoch (Jan 1, 1970 00:00:00 GMT) as a long integer, when the action was performed.
- user is the user name used to login, e.g, 'couch'.
- action is what the user did, e.g., 'login', 'logout', etc.
- status is 'success' or 'failure'.
- address is where the request originated.

The output schema should be: `out: {user: chararray, times:int}` where

- user is the username as before.
- times is the number of times the user did a 'login' from a particular address, without a matching 'logout' afterward.
- the output is sorted in order of laziness: most failed logouts first.